Profibus DP Master for PC

Pavel Trnka[1] and Petr Smolík[2]

[1] trnkap@control.felk.cvut.cz
[2] xsmolikp@control.felk.cvut.cz

Department of Control Engineering
Faculty of Electrical Engineering
Czech Technical University in Prague

Abstract.

This work implements industrial network protocol Profibus DP for master devices. It is targeted to computers lacking any Profibus communication hardware support. It allows for connecting Profibus to a common PC UART or a PCI-based RS-232/485 card and makes Profibus access available under Windows NT/2000/XP. Solved problems incorporated generating very short timing intervals under Windows (as short as 1μs), which are necessary for implementing fast communication protocols. These short intervals are achieved by writing the code as kernel device driver with maximal utilization of available hardware possibilities.

Keywords: Profibus DP, Real-time in Windows, RS–485

Introduction

Profibus is the most widely used communication bus (fieldbus) in the industry. This work is concerned in its variation *Profibus DP* (Distributed Peripherals) designed for communication between control units (like PLC or industrial computers) and distributed peripherals (sensors, actuators) over shared connection.

Profibus is master-slave protocol with deterministic medium access control (stations can transmit only with permission), where one connection can be used by multiple masters and each master has granted maximum time delay for acquisition of the transmit permission. Physical layer can be based on RS–485 standard and maximal baud rate is 12Mbps, which is fairly high for industry bus.

Stations can be of two kinds. Master stations allowed to initialize communication after receive of the permission and slave stations waiting for requests from master stations. Multiple master stations (control units) sequentially exchange permission to transmit called Token and thus forming a logical token ring.

With arising use of PCs' for control purposes in the industry, Profibus access is also demanded from them. So far it was necessary to have special hardware card, which was able to handle Profibus access. These cards are based on their own communication processor, customer's circuit or another hardware solution with enough power to meet high Profibus requirements.

Disadvantage of these Profibus cards is their high price. This brought the motivation to make a realization of Profibus as cheap and simple as possible. To accomplish that, we made it possible to connect Profibus to standard serial port RS–232 or to simple PCI communication card based on UART[3] circuit. Profibus DP Master is created as software implementation and special features of hardware solutions are substituted by maximal utilization of common hardware resources in PC. This was mainly possible by creating Profibus DP Master as system driver for Windows NT/2000/XP.

Connecting Profibus to PC

RS–232 Serial Port

Profibus DP hardware layer is based on RS–485, therefore to connect it to common PC RS–232 serial port we have to use interface. For simple applications (few stations, short cables) we can use simple unpowered interface (Figure 1).

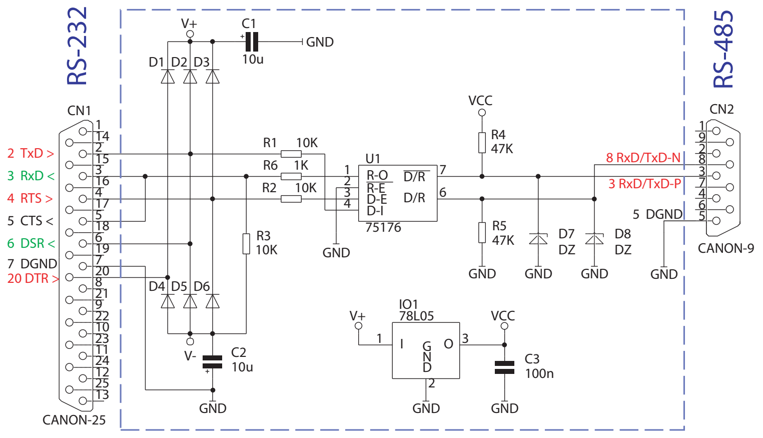

Fig. 1. Scheme of simple unpowered RS–232/485 interface.

Operation of this interface is quite simple. It is powered directly from serial port and the direction of data flow is switched by RS–232 output RTS. Before transmission, RTS is set to "1" and after completion of transmission RTS is set to "0" to allow receive of reply from the other station.

This is straightforward but the problem arises when we use this interface with PC RS–232 serial port and multi-tasking environment like Windows. Simply because UART doesn't signalize the end of the transmission (silence on the bus, the moment for switching the direction) by the interrupt, but it signalize it only by the flag in one of its status byte. The problem is that we can't continuously check this flag, but we need some interrupt. The UART has interrupt only for empty transmission buffer, but it comes before the end of transmission.

This was overcome by data loop-back in the interface. Direction switching is thus made after back receive of the last transmitted byte. In some special cases (timing without

[3] Universal Asynchronous Receiver/Transmitter

transmitting on the bus) we use connection between data output TxD and input DSR to generate modem interrupt as a signal of the end of the transmission.

PCI Serial Communication Cards

Baud rates over 19.2kbps are not achievable on PC RS–232 serial port, because available data rates (up to 115kbps) do not match data rates allowed by the Profibus specification [2]. For higher data rates we have to use communication card. In this work we used universal PCI communication card based on circuit OX16PCI954 [5] compatible with 16C950, which is able to communicate on all Profibus data rates (with proper crystal oscillator) and can generate interrupt for finished transmission. Moreover communication cards may already have RS–485 serial port.

Fast Timing in Windows

The important issue in implementing communication protocol is to assure proper timing according to protocol specification (correct time delays, keeping maximum possible intervals between frames, etc.), mainly because failure in timing yields to poor bus efficiency or even in its complete failure. Profibus is industrial protocol and thus stations needs real-time interaction, which relies on accurate timing.

Most important timing intervals for implementing Profibus are time-outs, which are necessary for granting maximum delay before acquisition of the transmit permission. What we need for implementing these time-outs is timing mechanism with sufficiently small time resolution. Required time resolution depends on current baud rate and for high speeds (Profibus maximum is 12Mbps) we need very small resolution nearly $1\mu s$.

Concerning implementation in Windows operating system we had to find the way for reaching this short timing intervals. Operating system offers timing functions such as `WaitForSingleObject`, but they all depend on the same principle allowing us to reach practically not less than about 10ms.

In this work we solved this problem by timing from UART interrupts, particulary interrupt signalizing empty transmission buffer. This makes it possible to achieve time resolution equivalent to the interval needed for transmitting of one byte, which is sufficient for Profibus implementation. Required interval is thus divided by the interval needed for transmitting of one byte and this yields to the number of bytes which must be transmitted before expiration of the interval.

But this incorporates the problem that we need UART both for access to the bus and for timing. This is solved by data flow direction control. When we need timing without transmitting to the bus we switch the direction to receive and all transmitted bytes are sent only to closed bus transceiver without any effect on the external bus.

Software Implementation

Profibus DP Master is written as Windows system kernel driver. It is mainly because we need to have direct hardware access (communication with UART, use of the interrupts), which is not available for user mode applications and is possible only in system drivers. Next advantage of realization as a driver is easier bus access for the user, because DP Master as a driver is integrated into the operating system.

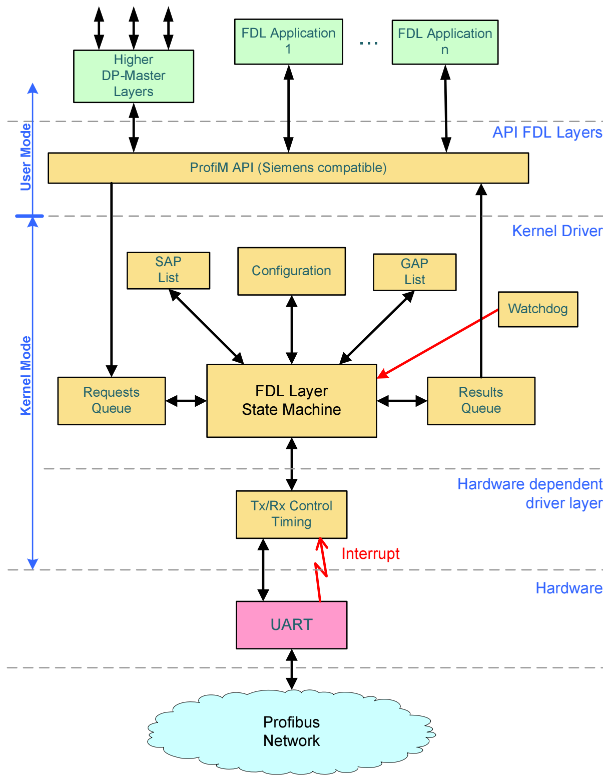

Fig. 2. Profibus DP Master Block Diagram

From the structure of DP Master in Figure 2 we can see that the driver running in kernel mode is divided in two parts. The first one is hardware dependent and directly controls UART circuit providing bus access (data flow direction control) and fast timing for higher layers. This is the only part, which has to be updated to allow driver to use a new hardware for Profibus access. Current version can use standard RS–232 serial port and PCI cards based on OX16PCI954 circuit.

The second part is hardware independent implementing DP Master and communication protocol and is included in the driver only to assure fast communication with lower layer to achieve fast access to the hardware.

The last part of DP Master is a static library linked to the programs using Profibus. This library provide program's access to the driver and Siemens compatible Profibus FDL API [4].

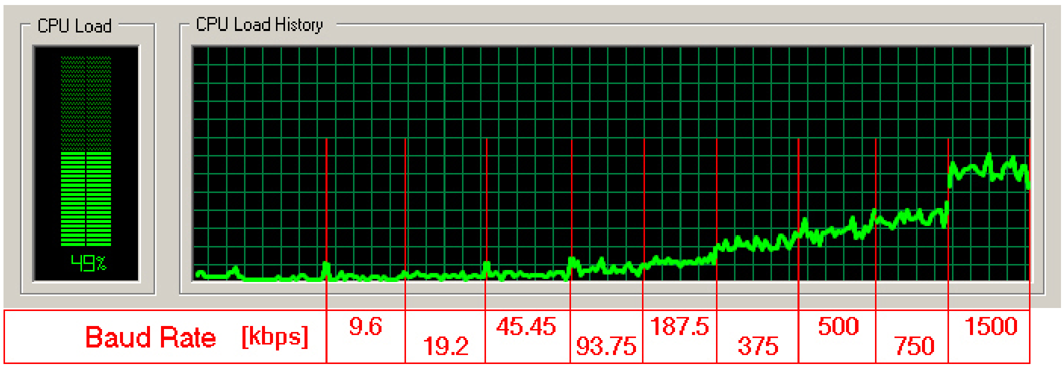

Fig. 3. Relation between CPU load and baud rate (measured on PC with AMD Athlon 1600+ using universal serial communication card Tedia PCI-1482 based on circuit OX16PCI954)

Practical Results

This work brings large simplification to the hardware required for Profibus access. On the other side the software realization increases CPU load in relation to baud rate as can be seen in Figure 3. This limits the usability of the software realization. In our case and on our testing machine the practical limit was 1.5Mbps (AMD Athlon 1600+ with universal serial communication card Tedia PCI-1482 based on circuit OX16PCI954).

Conclusion

This work tried to bring the solution demanded from the Profibus community. The effort was successful and the result can be used even for high transfer data rates in applications, which are not explicitly time critical. This was mainly possible by creating Profibus DP Master as system driver for Windows NT/2000/XP. Moreover application interface is compatible with solutions from Siemens company, which allows for simple replacement.

References

1. ONEY, WALTER. *Programming the Microsoft Windows Driver Model.* Second Edition. Microsoft Press, 2002. 800 p. ISBN 07-3561-803-8.
2. *PROFIBUS Specification.* Edition 1.0. Karlsruhe: PROFIBUS International, 1997. 924 p. European Standard EN 50 170.
3. TRNKA, Pavel. *Profibus DP Master pro PC.* Master Thesis. Prague: Czech Technical University, Faculty of Electrical Engineering, Department of Control Engineering, 2004. 70 p. URL:<http://dce.felk.cvut.cz/dolezilkova/diplomky>
4. *FDL Programming Interface.* Release 4. Karlsruhe: Siemens AG, 1995. 126 p.
5. *OX16PCI954 Data Sheet.* Revision 1.3. Oxfordshire: Oxford Semiconductor LTD., 1999. 72 p. URL: <http://www.oxsemi.com>